Table des matières

[1. Présentation 3](#_Toc154058879)

[1.1. Description générale 3](#_Toc154058880)

[1.2. Règles du jeu 3](#_Toc154058881)

[1.3. Cinématique des écrans 3](#_Toc154058882)

[2. Conception – Diagramme de classe 4](#_Toc154058883)

[2.1. Présentation générale 4](#_Toc154058884)

[2.2. Présentation détaillée MainWindow 4](#_Toc154058885)

[2.3. Présentation détaillée autre classe 5](#_Toc154058886)

[3. Partie Algorithmie 5](#_Toc154058887)

[3.1. Détection des collisions tirs joueur sur ennemi 5](#_Toc154058888)

[4. Conception graphique 6](#_Toc154058889)

[5. Cahier de recettes 7](#_Toc154058890)

[5.1 Tests de validation 7](#_Toc154058891)

[5.2 Tests de performance 7](#_Toc154058892)

Attention : toute vos impressions écrans doivent être lisibles !

# Présentation

## Description générale

## 

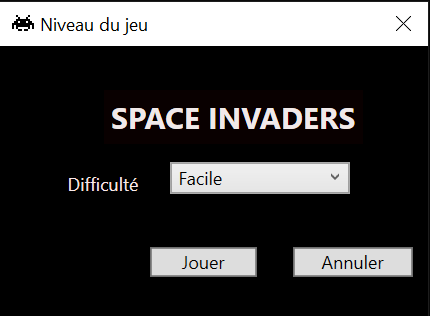
Genre et principe du jeu, but du joueur.

Impressions écran, avec des phrases introductives et explicatives.

## Règles du jeu

Règle du jeu détaillée ainsi que la description des touches ou autre nécessaire pour jouer (code triche par exemple …)

## Cinématique des écrans



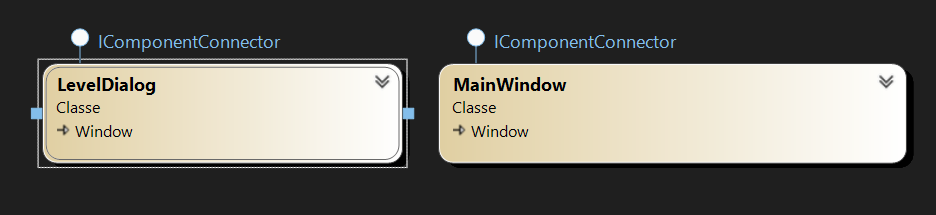


# Conception – Diagramme de classe

Attention : cette partie doit justifier et expliquer la note de l’autoévaluation

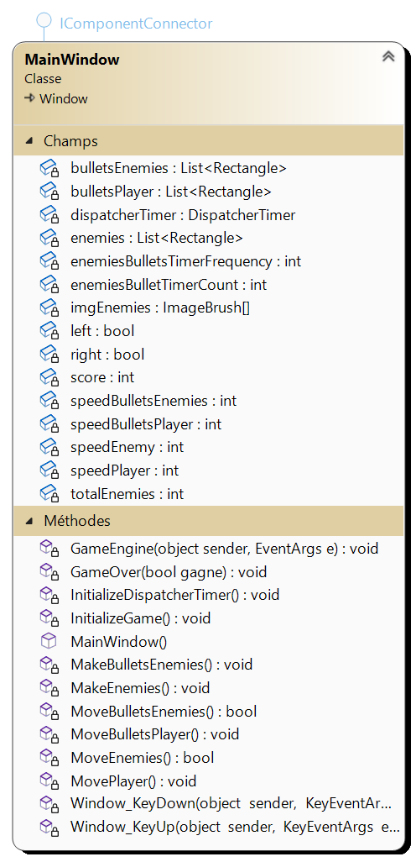
## Présentation générale

Faites une impression écran générale (sans le détail des classes ) accompagnée d’explications synthétiques pour chaque classe. N’hésitez pas à regrouper et découper si schéma trop grand.



Aide : sur votre projet, « Ajoutez un nouvel élément / Diagramme de classes » puis faites glisser vos classes depuis l’explorateur de solution dans la fenêtre.

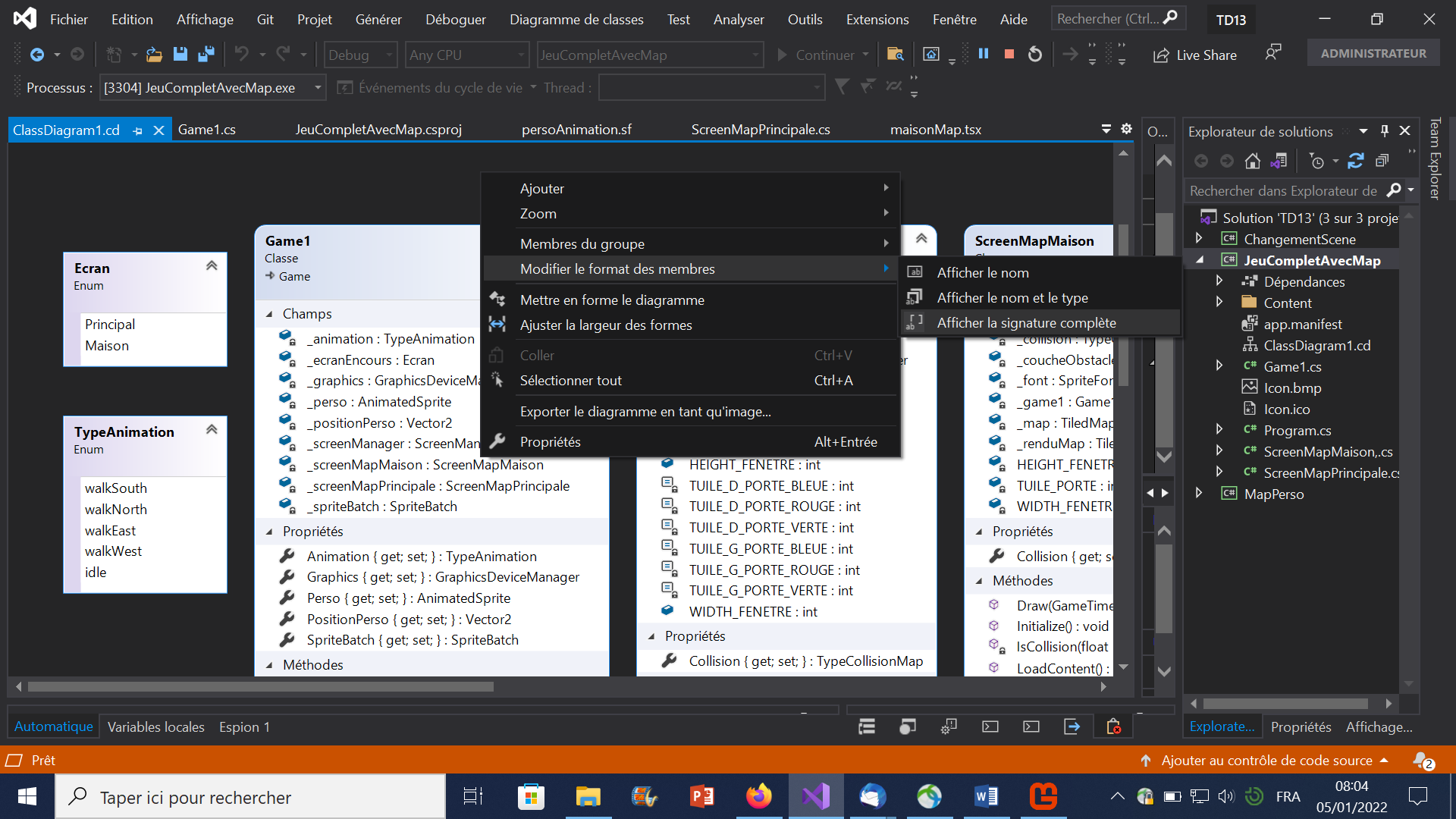
Justifiez vos choix de conception : expliquez et mettez en évidence si vous avez fait des classes pour améliorer, factoriser votre code



## Présentation détaillée MainWindow

Impression écran détaillée de chaque classe avec leur signatures **accompagnées d’explications un peu plus détaillées.**

Aide : clic droit sur le fond de fenêtre vide de votre diagramme de classes puis Menu ci-dessous :



## Présentation détaillée autre classe

….

# Partie Algorithmie

Attention : cette partie doit justifier et expliquer la note de l’autoévaluation

Listez les points d’algo intéressants (leur but et leur logique en pseudo-code voir exemple si dessous) à mettre en valeur pour démontrer la qualité de votre travail ( avec emplacement dans votre projet : Fichier et numéro de ligne ) Mettez en évidence leur complexité et accompagnez les d’affichage de debug pour justifier la validité de vos choix. Démontrez aussi le cout. Une sous partie pour chaque point important.

Remarque : Il est possible d’utiliser des algorithmes existants (ex : A\*) dans ce cas la expliquer la façon dont vous les avez utilisés/codés.

## Détection des collisions tirs joueur sur ennemi

On a décidé de tester tous les tirs avec tous les ennemis :

Pour chaque tir du joueur

Deplacement du tir

S’il dépasse de la zone de jeu, on prévoit de le supprimer

sinon

Pour chaque ennemi

si collision ennemi/joueur

on prévoit de ennemi et joueur

on marque un point

Finsi

Fin Pour

Finsinon

Fin pour

Dans MainWindow.xaml.cs : méthode MoveBulletsPlayer (ligne 243)

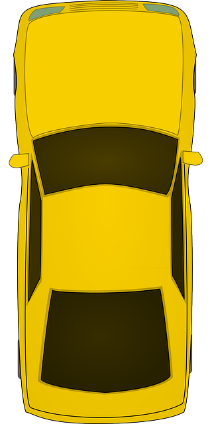
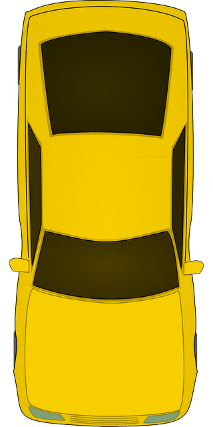
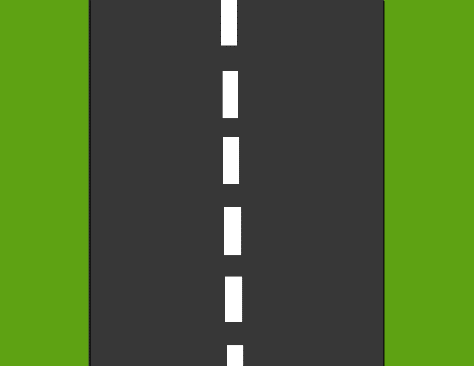
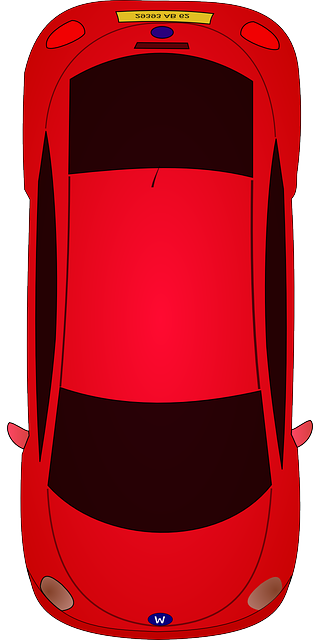
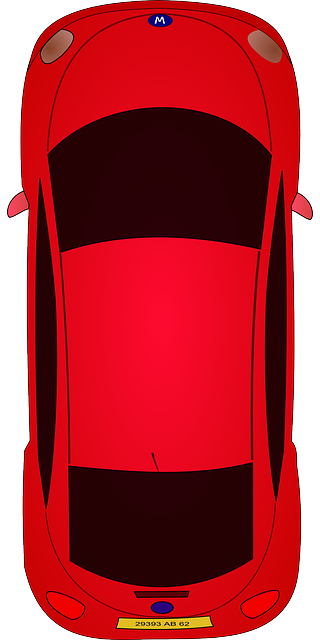
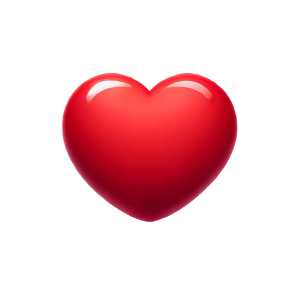
On peut compter 30 tests (au maximum : 1 par ennemi) pour chaque tir, à raison de 10 tirs max à gérer en simultané, on peut estimer à 300 tests toutes les 16 ms, soit 18 600 tests par seconde.

## Menu du jeu

Pour le menu, nous avions dans un premier temps créé une nouvelle fenêtre appelée « Menu », puis nous avions modifié le code dans l’App.xaml afin de forcer le démarrage de la fenêtre « Menu.xaml » avant celui de la « MainWindow.xaml ». Ensuite, en sélectionnant une des trois difficultés proposées, la fenêtre « Menu » se fermait et la « MainWindow » s’ouvrait.

Mais par la suite, nous nous sommes rendu compte qu’il existait une manière plus simple de procéder, en initialisant l’affichage de ce menu directement lors de l’initialisation de la MainWindow.

# Conception graphique

******

Toutes les images utilisées sont libres de droits, certaines récupérées sur Pixabay (https://pixabay.com/fr/) d’autres créées par l’IA. Nous avons nous-même créé la route. Pour l’escargot et les voitures, nous avons dû créer des images miroir pour pouvoir les retourner en fonction de la direction dans laquelle les éléments vont.

# Cahier de recettes

## 5.1 Tests de validation

Faites un listing des fonctionnalités , le nom de l’étudiant qui a travaillé dessus et son état :

|  |  |  |
| --- | --- | --- |
| Rimond | Création du menu | Etat |
| Rimond  Gastaldy | Recherche et création des images | OK |
| Rimond | Gestion des collisions | OK |
| Gastaldy | Déplacements de l’escargot | OK |
| Rimond | Déplacements des voitures, des salades et de la souris | OK |
| Gastaldy | Création et gestion du score | OK |
| Gastaldy | Ajout des grands-parents escargot et de la famille escargot | OK |
| Gastaldy | Gestion des barres de vie | OK |
| Gastaldy | Transport des salades vers les escargots à nourir | OK |
| Gastaldy | Gestion son/musique | OK |
| Rimond | Création et gestion des timers | OK mais peu optimisé |
| Gastaldy | Création du menu pause | OK |
| Rimond | Rédaction des règles | OK |
| Rimond | Gestion des respawn | OK |

Bien entendu, lors de problèmes importants rencontrés, nous nous mettions à deux sur le problème afin de le résoudre, peu importe ce sur quoi portait le problème.

## 5.2 Tests de performance

A l’aide des outils de diagnostiques : vous prendrez quelques mesures (captures écrans) à des moments clefs de votre jeu de l’utilisation de la mémoire et du processeur. Vous commenterez bien évidemment les moments choisis et les mesures.

